基于 0.618 法的改进区间插值法

宗钰涵*，张彦博

(北京邮电大学理学院，北京 100876)

摘 要：0.618 法是一维线搜索中针对一维单峰函数，应用最为广泛的一种方法。具有良好的收敛性，但其收敛性太慢，因此，本文基于函数在搜索区间端点和区间内任一点函数值的基础上，给出了一种普适性的线搜索加速策略，每步迭代都可以在较大程度上缩小函数值的不确定性区间。数值试验结果表明，其收敛速度较 0.618 法有所提高，尤其是当初始区间两端函数值相差较大或很大的情况下，本文改进算法可以很大程度上减小区间范围。

关键词：一维搜索；区间插值；0.618 法；加速策略

中图分类号:O224 文献标志码:A

An improved interval interpolation method based on 0.618 method

ZONG Yu-han*, ZHANG Yan-bo

(School of Science,Beijing University of Posts and Telecommunications,Beijing 100876,China)

Abstract: The 0.618 method is the most widely used method for one-dimensional unimodal function in one-dimensional line search.It has good convergence, but its convergence is too slow. Therefore, based on the search of function values at the end points of the interval and any point in the interval, this paper presents a universal line search acceleration strategy. Each iteration can reduce the uncertainty interval of function values to a large extent.Numerical results show that the convergence speed of the proposed algorithm is better than that of the 0.618 method, especially when the function values at both ends of the initial interval differ greatly or greatly, the improved algorithm in this paper can reduce the interval range to a large extent.

Key words: One-dimensional search; Interval interpolation; 0.618 method; To accelerate the strategy

0 引言

在最优化的迭代算法中，需要用到一维搜索或简称线搜索，这是许多非线性规划算法不可或缺的一部分。它可归结为单变量函数的极小化问题[1]，其数学模型为：$\min f(x)$。其分为精确线搜索和非精确线搜索[1]-[3]，而精确线搜索又分为直接法和解析法。解析法是使用函数导数的搜索，如插值法、牛顿法及抛物线法等；但当函数的导数不可用时，就需要采用不需要导数的方法，即直接法求解，其基本思想是：首先确定包括问题最优解的搜索空间，然后采用某种插值或分割技术缩小这个区间，进行精确求解。此类方法包括黄金分割法、分数法及二次插值等。在一维搜索中，一般要求函数是单峰函数，因此本文只考虑单峰函数情况。

现实中大多数函数的导数是不可用或极难求得的，这时候直接法的应用便广泛起来。从上述直接法的思想可以看出，决定直接法性能的关键有两点：一是初始区间的选取，二是包含极小值点子区间的缩小[4]。

0.618 法是一维直接搜索中应用最为广泛的一种方法，其优点是对函数的连续性不做要求；每次

迭代只计算一个新的试探点，另一个是重复利用的；具有良好的收敛性，收敛速度是线性的。但缺点是收敛太慢。本文在 0.618 法的基础之上，提出了一种改进的优化方法。其并将其与 0.618 法比较，通过数值计算证明了此种方法收敛速度快，迭代步数少。

1 0.618 法

0.618 法也称黄金分割法，其基本思想[5]是：通过对每步试探点函数值的比较，使包含极小值点的搜索区间（不确定区间）不断缩小，当达到某种准则时，可以找到极小值点的近似。

0.618 法取试探点的规则为：

$$\lambda_k = a_k + 0.382(b_k - a_k)$$

$$\mu_k = a_k + 0.618(b_k - a_k)$$

其计算步骤如下：

步骤 0： 给定初始区间 $[a_0, b_0]$ 及精度 $\varepsilon \in (0,1)$ 计算试探点 λ_1 和 μ_1，计算公式为

$$\lambda_1 = a_0 + 0.382(b_0 - a_0)$$

$$\mu_1 = a_0 + 0.618(b_0 - a_0)$$

计算函数值 $f(\lambda_1)$ 和 $f(\mu_1)$，令 $k=1$。

步骤 1： 若满足 $|b_k - a_k| \le \varepsilon$，则停止计算，得到极小值点的近似 $x^* = \dfrac{b_k + a_k}{2}$。否则，当 $f(\lambda_k) > f(\mu_k)$ 时，转步骤 2；当 $f(\lambda_k) \le f(\mu_k)$ 时，转步骤 3。

步骤 2： 置 $a_{k+1} = \lambda_k$，$b_{k+1} = b_k$，$\lambda_{k+1} = \mu_k$，

$$\mu_{k+1} = a_{k+1} + 0.618(b_{k+1} - a_{k+1}),$$

计算函数值 $f(\mu_{k+1})$，转步骤 4。

步骤 3： 置 $a_{k+1} = a_k$，$b_{k+1} = \mu_k$，$\mu_{k+1} = \lambda_k$，

$$\lambda_{k+1} = a_{k+1} + 0.382(b_{k+1} - a_{k+1}),$$

计算函数值 $f(\lambda_{k+1})$，转步骤 4。

步骤 4： 置 $k := k+1$，返回步骤 1。

2 基于区间插值的优化算法

文献[6][7]提出了基于区间缩小的不同改进算法，它们分别利用区间一端一阶导数和二阶导数，进一步缩小区间，但对于导数不可求或极难求得时，上述方法就不再适用。因此在本文中，我们希望通过利用当前搜索区间两端点的函数值及此区间上任意一点的函数值，构造出一种一维线搜索的加速算法，来改进 0.618 法。

设 $f(x)$ 为导数不可求或极难求得的一维连续单峰函数，$[a_k, b_k]$ 为包含极小值点 x^* 的当前搜索区间，首先我们选取 $f(a_k)$ 与 $f(b_k)$ 中函数值大的点，比如 $f(a_k) < f(b_k)$，则一定存在一点 ξ 满足

$$f(\xi) = f(a_k),\quad \xi \in [a_k, b_k]$$

上述属性表明区间 $[\xi, b_k]$ 不包含 x^*。所以我们可以在进行下一步前如先去掉区间 $[\xi, b_k]$，之后再对新区间进行分割，会使算法具有更好的效能。若想利用上述优点，关键在于如何求 ξ 使得 $f(\xi) = f(a_k)$。若所给问题函数简单，可以得到满足条件的精确解，但适用范围很局限，此时就需要另一种求解 ξ 的通用方法。

因为插值法是只利用函数值信息，不涉及导数信息，因此我们提出用插值法来求得可代替 ξ 的一个近似值 ς。具体思想为：利用函数在初始区间 $[a_0, b_0]$ 端点处的函数值 $f(a_0)$，$f(b_0)$ 及初始区间内任一点 $c \in (a_0, b_0)$（一般我们选此点为区间中点）的函数值 $f(c)$，来构造目标函数的二次逼近

函数$\phi(x)$，即利用三点二次插值方法构造逼近函数[8]，之后利用此函数求解满足$\phi(\varsigma)=\phi(a_k)$的$\varsigma$。

具体过程[3]为：通过曲线$y=f(x)$的三点(x_1,f_1)，(x_2,f_2)，(x_3,f_3)作二次函数$y=\phi(x)$，即

$$\phi(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f_1+\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f_0+\cdots$$

令$\phi(\varsigma)=\phi(a_k)$，得

$$\varsigma=\frac{\left(f(a_k)\left(c_k{}^2-b_k{}^2\right)+f(b_k)\left(a_k{}^2-c_k{}^2\right)+f(c_k)\left(b_k^2-a_k^2\right)+a_k\left(f(a_k\cdots\right.\right.}{f(b_k)(a_k-c_k)+f(a_k)(c_k-b_k)+\cdots}$$

对于$\phi(\varsigma)=\phi(b_k)$，类同。

下面给出此改进方法的算法步骤：

步骤0：给定初始搜索区间$[a_0,b_0]$和区间内一点$c_0=\frac{a_0+b_0}{2}$及精度$\varepsilon\in(0,1)$。计算函数值$f(a_0)$，$f(b_0)$，$f(c_0)$。置$k=0$。

步骤1：若满足$|b_k-a_k|\le\varepsilon$，则停止计算，得到极小值点的近似$x^*=\frac{b_k+a_k}{2}$。否则转步骤2。

步骤2：若$f(a_k)<f(b_k)$，按公式

$$\varsigma=\frac{\left(f(a_k)\left(c_k{}^2-b_k{}^2\right)+f(b_k)\left(a_k{}^2-c_k{}^2\right)+f(c_k)\left(b_k^2-a_k^2\right)+a_k\left(f(a_k)(b_k-c_k)+f(b_k)(c_k-a_k)+f(c_k)(a_k-b_k)\right)\right)}{f(b_k)(a_k-c_k)+f(a_k)(c_k-b_k)+f(c_k)(b_k-a_k)}$$

计算ς，去掉区间$[\varsigma,b_k]$，置$b_k=\varsigma$得新的区间$[a_k,b_k]$，转步骤4。否则转步骤3。

步骤3：当$f(a_k)\ge f(b_k)$时，按公式

$$\varsigma=\frac{\left(f(a_k)\left(c_k{}^2-b_k{}^2\right)+f(b_k)\left(a_k{}^2-c_k{}^2\right)+f(c_k)\left(b_k^2-a_k^2\right)+b_k\left(f(a_k)(b_k-c_k)+f(b_k)(c_k-a_k)+f(c_k)(a_k-b_k)\right)\right)}{f(b_k)(a_k-c_k)+f(a_k)(c_k-b_k)+f(c_k)(b_k-a_k)}$$

计算ς，去掉区间$[a_k,\varsigma]$，置$a_k=\varsigma$得新的区间$[a_k,b_k]$，转步骤4。

步骤4：计算试探点λ_k和μ_k，公式为

$$\lambda_k=a_k+0.382(b_k-a_k)$$

$$\mu_k=a_k+0.618(b_k-a_k)$$

计算函数值$f(\lambda_k)$和$f(\mu_k)$。当$f(\lambda_k)>f(\mu_k)$时，转步骤5；当$f(\lambda_k)\le f(\mu_k)$时，转步骤6。

步骤5：置$a_{k+1}=\lambda_k$，$b_{k+1}=b_k$，$c_{k+1}=\mu_k$，转步骤7。

步骤6：置$a_{k+1}=a_k$，$b_{k+1}=\mu_k$，$c_{k+1}=\lambda_k$，转步骤7。

步骤7：置$k:=k+1$，返回步骤1。

3 数值实验

为验证改进算法的效果，我们选取一个简单的一维单峰函数和两个较复杂的单峰函数[9]来比较改进的算法与0.618法，将它们对例1、2、3中目标函数在指定区间进行搜索，通过对比两算法的迭代步数和输出结果精度，衡量新算法的优势。其中精度我们分别选取10^{-2}，10^{-4}，结果如表2、表3、表4所示。

例1：$f(x)=x^2$，最小值点为0，初始区间选为[-1,100].

例2：$f(x)=e^x-3x^2$，最小值点为-0.144275，初始区间选为[-1,4].

例3：$f(x)=\cos x+(x-2)^2$，最小值点为2.35424，初始区间选为[1,4].

表1　函数1在两种算法下的结果比较

Table 1 Comparison of the results of function 1 under the two algorithms

	改进算法	0.618法
初始区间	[-1,100]	[-1,100]

迭代步数	6	20
输出结果	1.15431×10^{-3}	2.11656×10^{-3}
初始区间	[-1, 100]	[-1, 100]
迭代步数	10	29
输出结果	-9.13146×10^{-6}	7.23312×10^{-5}

表 2　函数 2 在两种算法下的结果比较

Table 2 Comparison of the results of function 2 under the two algorithms

	改进算法	0.618 法
初始区间	[-1, 4]	[-1, 4]
迭代步数	5	10
输出结果	-0.147635	-0.144297
初始区间	[-1, 4]	[-1, 4]
迭代步数	13	23
输出结果	-0.148459	-0.144314

表 3　函数 3 在两种算法下的结果比较

Table 3 Comparison of the results of function 3 under the two algorithms

	改进算法	0.618 法
初始区间	[1, 4]	[1, 4]
迭代步数	6	11
输出结果	2.35115	2.3542
初始区间	[1, 4]	[1, 4]
迭代步数	12	22
输出结果	2.3468	2.3542

通过观察三个函数的测试结果，改进算法的迭代步数不仅比 0.618 法的要小得多，而且最优值点精度也有明显改进，同时构造出的点列能够在有限步内得到达到所要求精度的最优化问题的最优解。因此改进算法不仅有稳定的收敛性，而且在各种情况下也能保持较快的收敛速度。

同时经过分析可知，0.618 法每步迭代只计算一个函数值，而改进的算法每步计算两个函数值，但因为迭代步数减少不止一半，因此改进算法也是有一定提高的。我们通过测试函数说明这一结论，针对例 1 在不同长度的区间上的区间缩短率，对 0.618 法和改进算法进行比较，结果如表 4 所示。

表 4　两算法对于不同区间的计算

Table 4 Two algorithms for different interval calculation

精度：10^{-4} 初始区间	0.618 法		改进算法	
	迭代步数	输出结果	迭代步数	输出结果
[-1, 500]	33	-3.96339×10^{-5}	10	-9.13188×10
[-1, 100]	29	-7.23312×10^{-5}	10	-9.13146×10
[-1, 10]	25	-2.76438×10^{-5}	10	-9.08866×10
[-1, 1]	21	-4.40305×10^{-5}	10	-8.88847×10
[-1, 0.1]	20	-3.79423×10^{-5}	8	-6.25798×10
[-1, 0.01]	20	-4.57181×10^{-5}	5	-1.86693×10

观察表 4，当初始区间是[-1, 500]时，迭代步数是 0.618 法的三分之一倍，同时区间缩短率减小了 28%；当初始区间是[-1, 0.01]时，迭代步数是 0.618 法的四分之一倍，同时区间缩短率减小了 36%；而当初始区间是[-1, 1]时，迭代步数虽然是 0.618 法的五分之二倍，但是区间缩短率只减小了 6%。因此我们可以发现当初始区间两端函数值相差较大或很大的情况下，比如初始区间两端点相差 100 倍左右时，改进的算法可以较快的减小区间范围，以较快的速度收敛到最小值点。

当区间长度慢慢变小时，0.618 法的迭代步数变化很小，或者几乎不变，但是相比较而言，改进的算法在区间长度减小时，不会出现停滞的现象，且迭代步数还会偶尔成倍减小，因此改进的算法可以显现出很好的性能。

4 结语

线搜索算法是单因素实验设计中最常用的一种方法，其收敛速度直接影响算法的效率。如何利用函数在搜索区间的信息，提高算法的收敛速度是相关研究的重点。本文通过利用目标函数在搜索区间端点处的函数值信息，不断缩小每步迭代区间，获得了具有普适性的加速策略，进行的数值实验表明本研究所给出的算法能够加快现有线搜索算法的收敛速度。

参考文献：

[1]陈宝林.最优化理论与算法[M]. 北京:清华大学出版社,2005: 254-280

[2]许国根,赵后随,黄智勇.最优化方法及其MATLAB实现[M].北京：北京航空航天大学出版社,2018：12-18
[3]张烨 . 基于混合遗传算法的离散平衡配重优化[J]. 电子元器件与信息技术 ,2021,5(1):116-117
[4]胡梦英, 贺祖国. 基于重开始共轭思想的改进多项式插值法[J].软件, 2015, 36（11）: 48-51
[5]李学文,闫桂峰,李庆娜.最优化方法[M]. 北京:北京理工大学出版社,2018：101-118
[6]姚胜伟,邬玉萍.一种线搜索加速策略及其应用[J]. 河池学院学报,2019,39(2)：43-48
[7]董媛媛,谢承蓉.一种改进的0.618法[J]. 长江大学学报,2014,11(22)：4-6
[8]林青瑜,洪智勇.对基于最优化原理的高压配电网建设规模评估分析[J]. 电子元器件与信息技术 ,2020,4(10):72-73
[9]Emin Kahya. A new unidimensional search method for optimization:The 5/9 method[J].Appl.Math.Comput,2005：163-179

作者简介：
宗钰涵(1997-)，女，硕士研究生，河北省，主要研究方向：最优化算法，379508292@qq.com。
张彦博(2000-)，男，本科生，主要研究方向：计算机。